The search process comprises a process for binding the variables, defined as follows, which makes it possible to determine the various variables of the identifiers Idi. Let Id be an object identifier in which the variables {W1, ...Wk} belong to the set {V1, ..., Vn} mentioned above. The identifier Id has the form a1.W1.a2.W2. ... .ak.Wk.ak+1, in which the coefficients ai have the structure of an object name in the ASN.1 sense of the term, which means that the coefficients ai are constituted by a sequence of positive integers.

Let j be the index for which the variables with successive indices {W1, ..., Wj} are already bound. Finding all of the objects that can be designated by this identifier consists of navigating through the instantiated MIB, searching for the objects by launching a request of the SNMP "GetNextRequest" type, using as a parameter an object identifier whose identifiers begin with the root R = a1.W1.a2.W2. ... .aj.Wj and by applying an object identifier unifiability criterion, defined below. Let Id' be the response to the SNMP request GetNextRequest applied to the root R, if such an object exists. The object does not exist if the response to the request GetNextRequest indicates that the designated object does not exist, in which case the search is terminated.

If Id and Id' are unifiable according to the criterion defined below, then the variables {W1, ..., Wk} are all bound, which means that they all have a value, and we have just found the identifier of an object that can be designated by Id, namely Id'.

In order to find the next valid combination of the variables {W1, ..., Wk}, knowing Id', it is necessary to reiterate the preceding process, using Id' as the starting point for the search, after having unbound, i.e., retrieved the previous values of, the variables Wi that were not bound at the very beginning of the preceding search process.

If Id and Id' are not unifiable and the identifier Id' begins with R, then the search continues as before, using Id' as the starting point and unbinding any variable Wi that was not bound at the very beginning of the preceding search process.

If Id' does not begin with R, then there is no longer any existing object whose identifier could be unified with Id. The search process is then terminated.

The unifiability criterion is defined as follows. Let Id' be a reference identifier (without a variable part) wherein all the elements are integer values; two identifiers Id and Id' are unifiable if both identifiers are the same size and if the identifiers Id.ai = Id'.ai for any i belonging to [1, k+1]. Let {w1, ..., wk} be the integer values of the identifier Id' that correspond to the variables {W1, ..., Wk} of Id, i.e. that are located at the same position in the corresponding identifier. The binding of the variables Wi takes place sequentially for indices varying between 1 and k through the following method:

if Wi is bound, i.e., if Wi already has a value, then this value must be equal to wi; if the latter condition is not satisfied, then Id and Id' are not unifiable and the binding of the variables {W1, ...Wk} is not possible;

if Wi is free, i.e., if Wi does not yet have a value, then the variable Wi will be bound to the value wi.

For example, let Id = 1.2.3.W1.4.5.W2.6.7 and Id' = 1.2.3.10.4.5.20.6.7. These two object identifiers are unifiable for variables of W1 and W2 whose values are 10 and 20. On the other hand, if Id = 1.2.3.W1.4.5.W1.6.7 and Id' = 1.2.3.10.4.5.20.6.7, then Id and Id' are not unifiable because the variable W1 cannot have the values 10 and 20 at the same time.

The identifier unification process makes it possible both to verify that two identifiers are unifiable and to determine for which values of the variables that are still free prior to the start of the unification process this unification is possible.

When the name resolution process has exhausted all the valid combinations, the indicator deployment agent (ADI) becomes unnecessary if the monitored configuration is never changed.

When the software and/or hardware configuration of the monitored machine can change, for example, according to the invention, when the predetermined maximum number of machines per subdomain has been reached, the indicator deployment agents are reactivated in order to re-evaluate the hardware and software configuration by performing a search for the instantiable elements, instantiating only those of the indicator agents that don't already exist, and deleting those that no longer have any reason to exist.

In a variant of embodiment, when the programming language is Java, the creation of a configuration agent, an indicator deployment agent or an indicator agent consists of sending an agent creation request "agentCreateRequest" to the "factory" agent of the agent machine that manages the subdomain in which the agent must be deployed. This request includes as parameters the identification of the agent and the status of the agent. The status of the agent corresponds to the serialized Java object of this agent. When the "factory" agent receives this request, it deserializes the serialized Java object and thereby obtains the requested agent.

It should be clear to those skilled in the art that the present invention allows for embodiments in many other specific forms without going beyond the scope of application of the invention as claimed. Consequently, the present embodiments should be considered as examples, but can be modified within the range defined by the scope of the attached claims, and the invention should not be limited to the details given above.